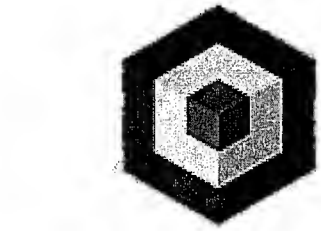
EXHIBIT 1

(Billing System Integration Technical Specification Design Documentation
Inktomi Corporation)

Billing System Integration Technical Specification
Design Documentation
Inktomi Corporation

Confidential



Inktomi®

Revision History

Date	Description	Author
1/5/2001	Initial Version	Pei-yuan Zhou & Philippe le Rohelec
1/11/2001	Minor edition	Pei-yuan Zhou
1/12/2001	CDS section added	Sailendra Padala
1/13/2001	Reformat and cleanup	Rene Chan

Table of Contents

I. Introduction 4
A. Purpose/Problem 4
B. Solution 4
II. Requirements 4
III. High Level Design 4
IV. Block Design 5
V. Technical Details 7
A. Portal Infranet configurations and customizations 7
1. Overview 7
2. Infranet Custom Objects Setup 8
3. List of groups Setup in Infranet 8
4. Infranet Custom objects Update 10
5. Custom Objects Loading in Infranet 11
6. Parse and filter billable events to Infranet through UEL (Universal Event Loader) 11
7. Sample Price List 12
B. Traffic Server related features 12
1. Retrieve a user identity (MSISDN) 12
2. Retrieve user profile and policies 13
3. Authorization based on user profile 14
4. Billing decision for transactions 15
5. Logging billable events 16
C. Other high priority features to support scalability and performance enhancement 16
1. Traffic Server Full Clustering support 16
2. Centralized billing event logging utilizing CDS Content Manager 16
3. The time periods of this log processing has to set appropriately taking into consideration of the UEL's performance, Infranet's performance, the log file sizes 16
3. and the file transfer latency from Traffic Servers to the CDS agent on the controller box. Billing event logging aggregation 17

I. Introduction

A. Purpose/Problem

Traditionally wireless voice service subscribers are billed based on duration of service used and whether roaming is involved. As a result, existing infrastructure only support duration based billing. However, a new way of billing wireless Internet access subscriber is needed. This is because the nature of Internet access is very different from placing a phone call. The subscribers are using the wireless handsets to access and interact with Internet content. Significant amount of time can be spent on simply logging onto the system. There can also be idle time when the user is "consuming" the requested data. Wireless Platform 1.0 needs to address this problem.

B. Solution

To effectively enable volume/content-based billing, integration is needed between the data delivery platform and the billing system. Every data access request will be authorized based on the subscriber's profile. Corresponding responses will be logged and sent back to the billing system.

The billing portion of the Wireless Platform 1.0 is mainly about integration between Portal's billing engine Infranet and Inktomi Traffic Server to enable authorization and billing of wireless Internet data services and applications.

II. Requirements

The billing system integration should enable content and volume-based billing.

While the target billing system of Wireless Platform 1.0 is Portal Software's Infranet, the platform should be extendable to handle different billing system with the addition of different billing system adapters.

The billing system integration should also allow configurable access control level for different users.

III.High Level Design

The billing tasks of the Wireless Platform 1.0 has two major portions:

- Portal Infranet configurations and customizations
- Traffic Server new plug-ins, configuration changes and package

Some major tasks in Portal Infranet configurations and customizations:

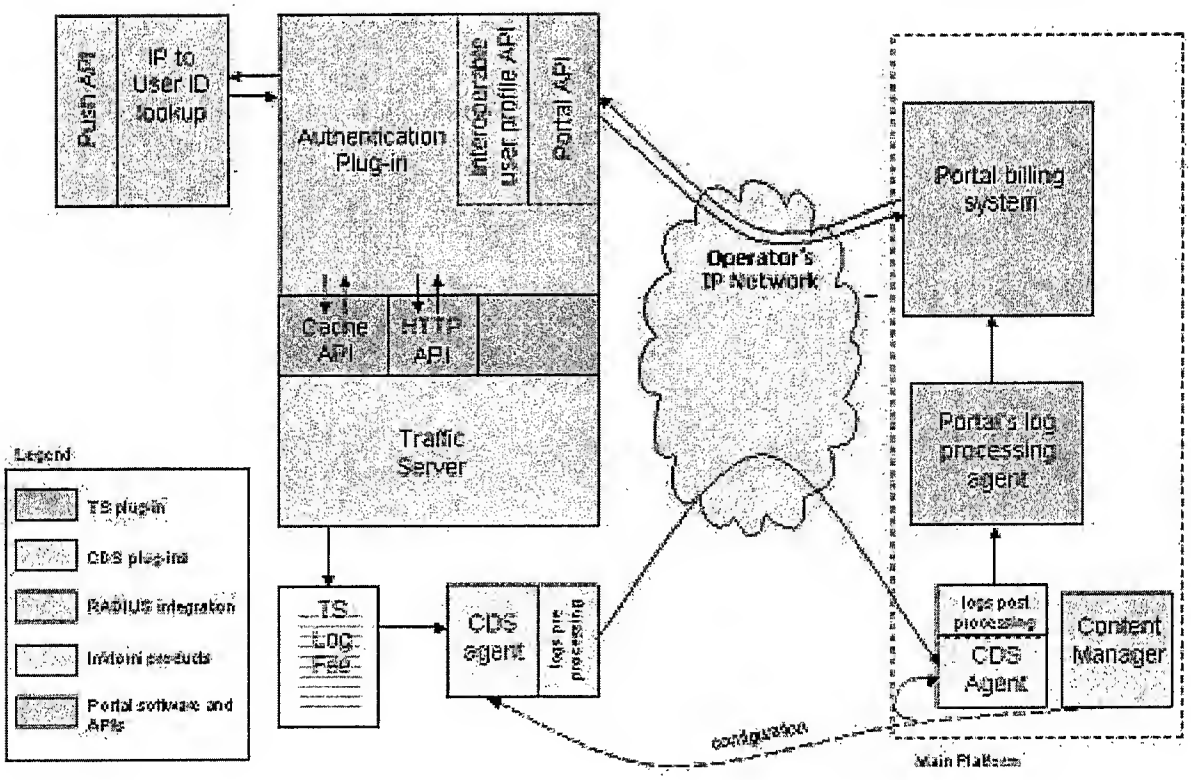
- Set up the Infranet custom object (URL groups and URL list)
- Update the Infranet custom objects

- Set up the list of groups in accounts
- Inject CDRs in Infranet (UEL)
- Retrieve custom objects from Infranet to Traffic Server plug-in
- Set up a sample price list

Other high priority features to support scalability and performance enhancement:

- Multiple Traffic Servers or Traffic Server Clustering support
- Centralized billing event logging utilizing CDS (Content Delivery Service)
- Billing event logging aggregation

IV. Block Design



Two Traffic Server plug-ins are added:

- Radius Server integration plug-in: receive a user session start/end information, find IP/User ID mapping and get user profiles.
- Authorization/Authentication plug-in: handle each HTTP transaction.

One or two CDS plug-ins are added:

- Log pre-processing plug-in: parse and send billing event log to the CDS server for further processing.
- Log post-processing plug-in: filter and send billing events to Infranet billing engine.

V. Technical Details

A. Portal Infranet configurations and customizations

1. Overview

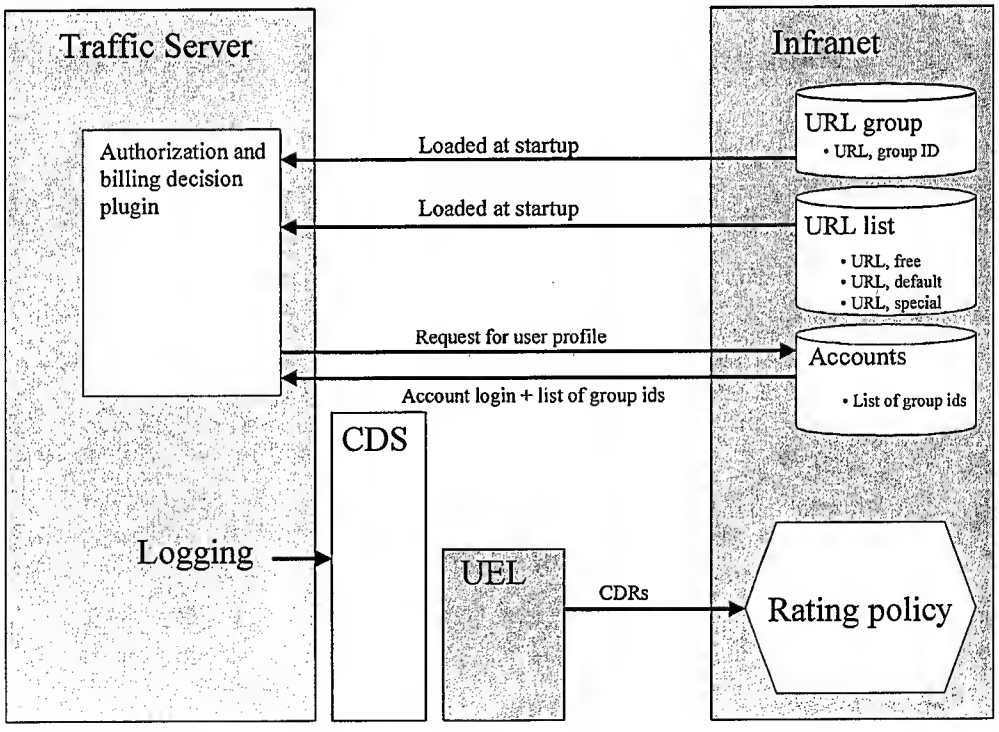
Events generated by Traffic Server are to be rated by the Infranet billing system.

Integrating with Infranet mainly consists of **formatting and sending the events** so that they can be used by Infranet.

Platform 1.0 also includes an **authorization** module that bans certain URLs from being seen by subsets of users. To be able to do that some data is needed about users and about URLs. This data is stored in Infranet. Groups of URLs are setup. Each user is linked to one or more group and can access any URL in any of these groups.

In addition, in order not to flood Infranet with CDRs, the events are **filtered** in platform 1.0. Filtering information is stored in Infranet. Filtering information is a list of URL, each of them being given an attribute that can be FREE, DEFAULT or SPECIAL.

At aggregation time CDS will remove all URL that are marked as free, group all the URL marked as default together and leave the URL marked as special as they are. Default URLs are URLs that are either marked as default or that are missing from the filtering information.



2. Infranet Custom Objects Setup

Using Infranet Pricing Tool and Administration Tool will accomplish most of required tasks.

We also need to create two special tables (custom Infranet objects) to store a given list of optional URLs.

The first table URL_LIST will store all URL with any billing impact. The columns are:

- URL String
- Name String
- Type Enum (free, default or special)

The TYPE field can have one of following three values:

- FREE: the site is FREE site
- SPECIAL: the site has special billing impact
- DEFAULT: the site is a common site as far as billing concerns, how to charge then depends on user policy

For any URL that does not match any URL in the table, the billing TYPE will be default.

The second table URL_GROUP will store all pre-grouped URLs as a list of optional services a subscriber can subscribe. If a subscriber subscribe a group, the group ID will be in the subscriber's user profile. If a user tries to access an unsubscribed group, the request will be denied at request time upon checking with the user profile. The columns in this table are:

- URL String
- Group_ID INT
- Group_Name String

One note is that since those objects are not standard Infranet objects, many things need to be considered. Among them:

- The URL's in URL_LIST table need to be consistent with Infranet billing setup
- The URL group ID in URL_GROUP table needs to be propagated into user profile setup
- Provision, described in more details in later sections

3. List of groups Setup in Infranet

a) Account Creation

Users will be created in Infranet the same way they usually are on every Infranet system. There is only one constraint:

All users must purchase a deal that will be included in platform 1.0 package. This deal gives access to URL groups and is defined in the price list. This deal is a list of group IDs.

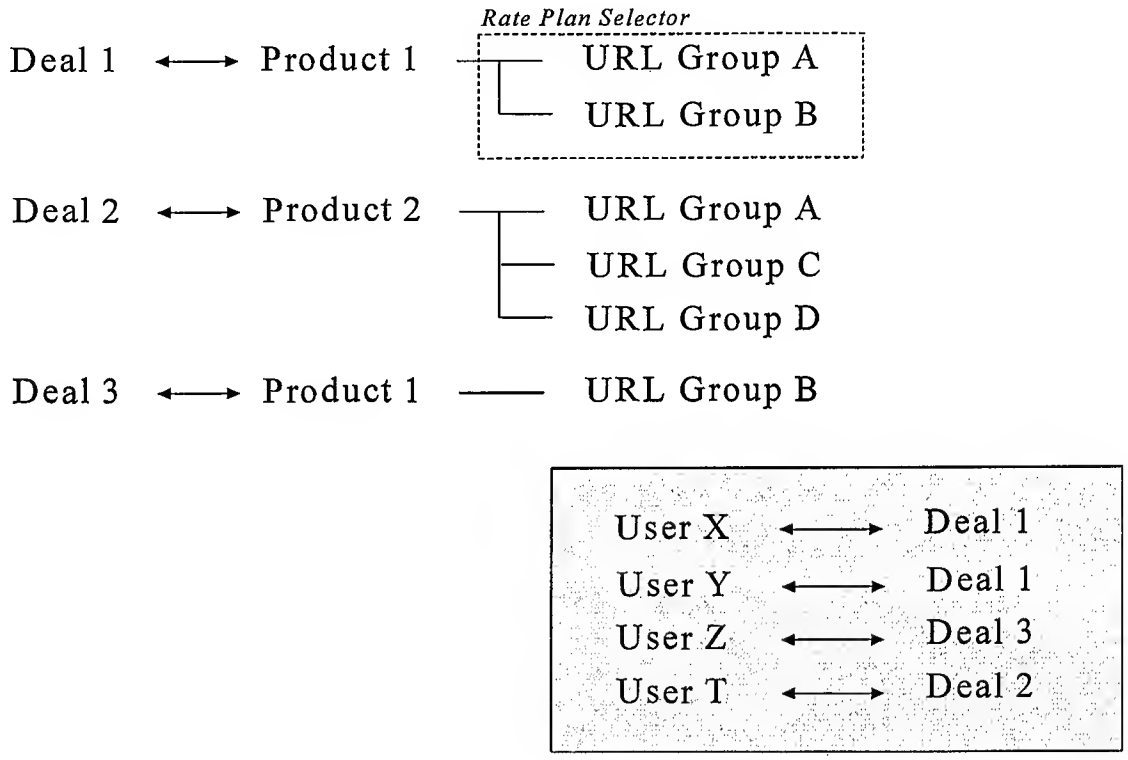
b) URL Group Deal

The link between accounts and URL groups is established in Infranet deals. Any updates on URL groups has to be reflected by the accounts, through setting up Infranet deals manually.

This deal is setup and updated using the Pricing Tool.

In our case, the deal includes one product. The product is a list of URL groups. The carrier must define as many different deals as he wants as combinations of URL groups.

Example:

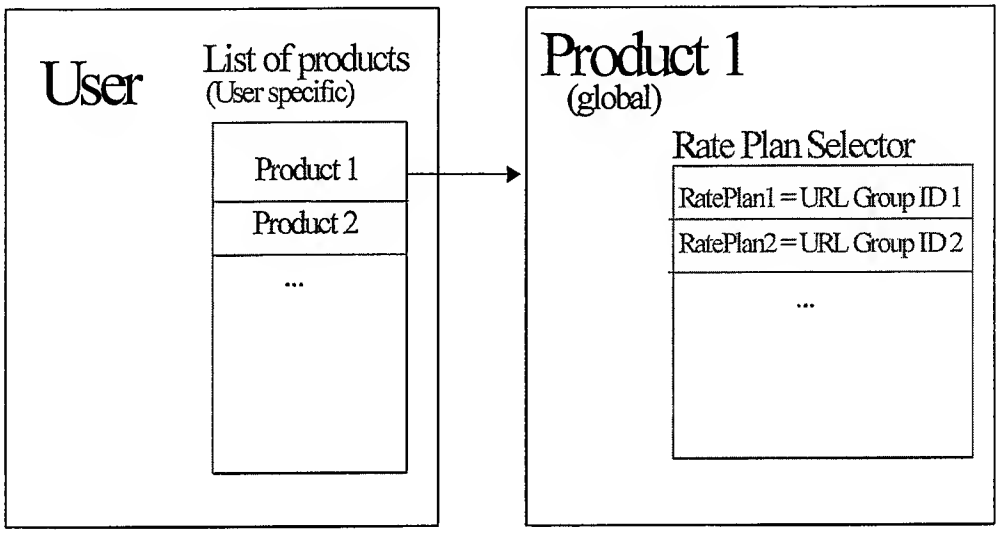


The link is a list of URL groups that is given to users. Lists of URL groups are stored in fake products in the price list. Those products are not actually used to rate events. Those products are placed in fake deals.

A rate plan selector is defined in the fake products. Each of the rate plans is a URL group ID. When an account is created, it must be given one and only one of the fake deals. The link is then established between the user and the groups of URL that he can use.

As products are global to all users, changing the list of rate plans in the rate plan selector of a given product (i.e. changing the list of URL groups) will affect all the accounts that have purchased this product.

Later on, if a change is needed in the list of URL groups for all users, the rate plan selector for the product simply needs to be modified. If a change is need only for one user, then the product must be cancelled for this user and another deal must be purchased.



4. Infranet Custom objects Update

a) URL Groups Update

The **URL_GROUP** table is updated using a batch process.

The batch process will check a command file once it started. The command file will have one command entry at each line with two columns:

- 1st one is the name of a text file including a list of URL
- 2nd is a group ID

Running the batch process, each line of command file will be processed one at a time. An error in one command or text file will only cause that command to fail, but not the others. Upon finish, all the old URLs related to the group ID are removed and the new URLs coming from the text files are added to the URL_GROUP table. If a new URL group is created, Infranet deal(s) have to be updated to reflect the group. If a URL group is removed, deals containing the group have to be modified as well.

Once the URL_GROUP table has been changed, access rights to URLs for all users are modified.

b) URL list – Billing Decision

The **URL_LIST** table is updated using a batch process. The batch process will look for a given name of a text file, which contains the new list of pairs of URL/billing types.

Running the batch, all the old URLs are removed and the new URLs coming from the text files are added to the URL_LIST table along with their type.

Special cautions will be provided so that old list will not be removed if the new list update is not successful or the new list is not complete.

Any URL in Infranet pricing tool should be imported into the URL_LIST table.

Once the URL_LIST table has been updated and has been reloaded in Traffic Server, the log generation process will use the new data.

5. Custom Objects Loading in Infranet

The URLs groups will be loaded in Traffic Server at startup and will be updated regularly.

The URLs list will be loaded in Traffic Server at startup and will be updated regularly.

The user profile will be retrieved from Infranet each time a request originated from this user comes into Traffic Server. The user profile consists of the user login + list of groups.

Loading data from Infranet is done using the Portal PCM API. It can be run from any machine having a TCP connection with an Infranet CM.

All the requests to Infranet will be done from the authorization plug-in.

6. Parse and filter billable events to Infranet through UEL (Universal Event Loader)

The Universal Event Loader is a client tool provided by Portal.

It is Java code and can be run from any machine having a TCP connection with an Infranet CM.

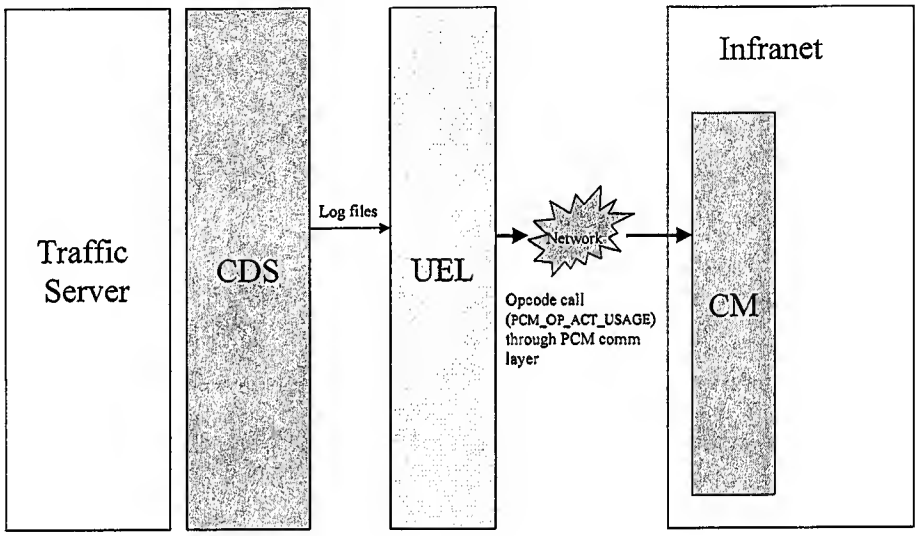
The UEL reads a text file filled with event logs and injects those events into Infranet for rating.

To be able to interpret the log file, the UEL needs a template that says which field in the log file should be mapped to which Infranet field.

The template is built using the Universal Event Mapper and stored in the Infranet database.

As the mapping is dependent on the structure of the price list, it is up to the users to define their own templates in accordance to his pricing policy.

Platform 1.0 will include a default price list that will take advantage of all the information given in the log file.



7. Sample Price List

The default price list specifies different rates according to:

- URL
- Media/Content type
- Number of bytes

Some events will be rated according to the 3 criteria (they are special URLs that need a specific pricing) and some others will be rated only according to the media type and the bytes number (they are default URLs).

B. Traffic Server related features

1. Retrieve a user identity (MSISDN)

This component interfaces with a RADIUS authentication server or with Network Access Servers (NAS) such as dial-in servers or GGSNs. On a user logon event, the component receives the user-id to and the IP address assigned to that user by the RADIUS server, and stores it in a table. On a logoff event for that user, the entry is removed for the table.

The component interfaces with a wireless network component, which may be a RADIUS server or with Network Access Servers (NAS) such as dial-in servers or GGSNs. We expect these components to have an interface to provide the logon and logoff events, however, these interfaces are not defined in a standard protocol (i.e. RADIUS), and therefore will vary from vendor to vendor, or may not exist at all.

Specific tasks:

1. A simple push protocol to receive the information from the RADIUS or from the NAS.

2. A default implementation as a Traffic Server plug-in to receive the information when a user logs on or logs off. The plug-in will have a blocking listen port to receive user session messages.

3. Customization in the form of a plug-in on the RADIUS server or the NAS to send the information to Traffic Server. This component can be customized for different network deployments.

4. A mechanism to store the IP/User ID mapping persistently during a user session

5. An API to the component allows the authorization plug-in to lookup the user-id, such as the MSISDN of the wireless subscriber, given its source IP address. The user-id is the key for lookup in the user profile database, and therefore required by the authorization plug-in.

The items 1 & 2 will be developed as TS plug-in. The plug-in will be in a separate thread.

The solution for 4 & 5 is to utilize the cache library of Traffic Server. The new TS cache can persistently hold the mapping during a user session; this solution can provide support of Full Traffic Server clustering. The first component or plug-in that retrieves the mapping between IP and the user name will insert it into TS cache with following features:

- The entry will be in HTTP format.
- It is configured to stay in cache infinitely or a configurable long duration.
- The cache key will be the URL of a GET HTTP request on the local host, i.e. //<local host>/<IP address>

There are two ways to retrieve the mapping from the cache. The first way is to use generic cache APIs. All Traffic Server plug-ins will use the approach. The second way is to use an HTTP GET request. Processes other than Traffic Server will issue a GET HTTP request to a Traffic Server to get the mapping.

The cache entry will be removed only when a user session end message is received from the RADIUS plug-in. There will be a leak in disk space if such a message does not forward to Traffic Server correctly. To prevent the disk leak problem, the time-to-live parameter associated with cache API should not be set forever.

2. Retrieve user profile and policies

In the Platform 1.0 release, a user profile only consists of a list of allowed site groups against a list of global optional groups. A user can access any other sites other than those in the optional groups. The retrieval of the user profile utilizes Portal Infranet API to retrieve the user related Account/Service/Profile information from Infranet.

The retrieval of user profiles can be clearly divided into two distinct tasks:

- Retrieve the list of global optional groups. The list of those groups are assigned by a carrier and incorporated into user plans and services. Each user can subscribe one or many of those groups. But they do not belong to any particular subscriber. At the initialization stage of Traffic Server, a global callback will retrieve those groups into TS cache. Each entry will consist of exact one URL, which is also the key of the entry. The entry also contains the group ID the URL belongs to.

- Retrieve an individual user subscription profile. It can happen in two cases:

    1. After a user logon happens

    2. An HTTP request comes. After DNS look up finishes, a transactional callback will check the user profile in cache. If it is a miss, then the user profile needs to be retrieved from Portal Infranet.

Once the user profile is retrieved, it will be saved in TS cache, and keyed by the user ID (MSISDN). The user profile will be removed when the user session ends.

The group list will never be purged. A timer-based handler will reload the group list at a given interval. Infranet API will be used to retrieve the group lists.

3. Authorization based on user profile

The sequence of authorization follows:

- For every HTTP request the plug-in is called at an HTTP state where the request header was read, and the DNS lookup for the content server is completed, so that the request header from the user-agent is accessible by the plug-in. A user HTTP request comes, the user identity is likely already known and is inserted into the HTTP header by WAP/HTTP protocol processing component.

- If the user identity is missing from the HTTP header, the user identity is retrieved from cache using cache API, then inserted into HTTP header as user name.

- If no user profile is found for the given user, the request is denied.

- Check the request URL with pre-retrieved URL group list, as well as the group ID the URL belong to.

- Search the cached user profile to see if the user has the group ID as the allowed group. If the user profile is not cached or if the cached profile is stale, the plug-in calls the user profile API to retrieve the profile from the Portal database. A Portal API linked to the authorization plug-in retrieves the user profile from the Portal billing system.

- If there is a miss, that means the request URL is not one of the optional sites, the request is allowed.

- If there is a match, comparing the access policies described in the user profile, and the requested URL, the plug-in allows or denies access to the URL.

- If access is denied, the plug-in sets an HTTP response code of 401 (unauthorized), and directs the HTTP state machine to respond with an error.

- If access is allowed, the plug-in directs the HTTP state machine to proceed processing the request.

4. Billing decision for transactions

Billing decisions have to be made during the processing of a user HTTP request due to several reasons. The primary reason is to insert some instruments in the billing event log so that later on some billing event aggregation is possible. Billing decision can also help to reduce the amount of logging events since we do not have to log events unrelated to billing. Another consideration is to support real-time billing for future releases.

Two tasks needs to accomplish the billing decision for transactions:
- Pre-load all specified URL into cache. Every URL with any billing impact will be stored in the Infranet URL_LIST table, and be retrieved through a customized Infranet API. This will be done by a global callback at the initialization time, and later re-loaded by a timer-based callback.
- At the each HTTP transaction, compare the request URL with the pre-loaded URL list to see if it is billable or not.

There is an attribute TYPE associated with each pre-retrieved URL in cache. The value of the attribute is one of:
- FREE: the URL is explicitly set to be free.
- SPECIAL: the URL is explicitly set to be chargeable.
- DEFAULT: the URL will be charged by the default user plan

The billing decision can be implemented in the same callback function as the authorization callback.

The sequence of billing decision follows:

- For every HTTP request the plug-in is called at an HTTP state where the request header was read, and the DNS lookup for the content server is completed, so that the request header from the user-agent is accessible by the plug-in. A user HTTP request comes, the user identity is likely already known and is inserted into the HTTP header by WAP protocol processing component.

- If the user identity is missing from the HTTP header, the user identity is retrieved from cache using cache API, then inserted into HTTP header as user name.

- Try to match the "path" portion of the request URL with the pre-loaded URL lists

- If there is a match, get the URL TYPE parameter; insert the value of the parameter into HTTP header. The name of the HTTP field is "@BILL:".

- If there is a miss, that means the request URL is not one of the optional sites, insert value of "DEFAULT" into HTTP header with the field name of "@BILL:".

5. Logging billable events

An existing callback for logging will be modified to log billable events in the traffic server. The billable event log is separated from the Traffic Server main log by configuration. Every allowed transaction, except those marked as "FREE", will be logged as one entry in the log file.

The log format will consist with Traffic Server group development. Each entry will contains following information:

- User ID
- IP address (optional)
- Timestamp of the transaction start time
- Upload data volume
- Download data volume
- URL
- Media type (video, text ...)
- Billable flag (SPECIAL, DEFAULT)

C. Other high priority features to support scalability and performance enhancement

1. Traffic Server Full Clustering support

Platform 1.0 will support Traffic Server clustering. It will not support multiple Traffic Servers on different platform configurations.

By utilizing Traffic Server cache API, Traffic Server clustering will be able to access IP/User ID mapping, user profiles, as well as URL global lists.

2. Centralized billing event logging utilizing CDS Content Manager

CDS Content Manager will be having CDS agents on each Traffic Server talking to the CDS agent on the Controller Box. The CDS agents on the Traffic Servers will copy the custom format billing logs of traffic server to the CDS controller box through the CDS agent sitting on the Controller on a timely schedule as defined using the CDS Console. After all the logs for that time period have been received by the CDS Agent on the controller, the agent will filter the logs for Default and Special billing records (removing the Free records) and will aggregate the Default and Special billing records into two separate kinds of files of standard sizes as defined using the console and will place them into different folders. The CDS agent on the controller will then start the Universal Event Loader specifying the folders and the files that need to be processed into billing events. After the UEL is done processing the logs the CDS agent will archive the logs.

3. The time periods of this log processing has to set appropriately taking into consideration of the UEL's performance, Infranet's performance, the log file sizes

and the file transfer latency from Traffic Servers to the CDS agent on the controller box. Billing event logging aggregation

Billing event aggregation before sending to Infranet will greatly improve the billing performance, which may likely be one of performance bottlenecks.

The platform 1.0 will only do some simple aggregations as follows:

- If an event marked as SPECIAL, no aggregation will be done, the event will be sent to the Infranet directly.
- If an event marked as DEFAULT, We will summarize it with other default events based on URL the user identity and the media type. And send the summarized result as one event to Infranet.

The aggregation process will be done using the CDS Content Manager.